Tissue Segmentation and Nuclei Classification

1. Dataset

For this task we use a subset of the PUMA dataset [8]. The dataset was divided into train, validation, and test splits with 163, 20, and 22 images respectively. The dataset consists of H&E stained melanoma regions of interests and has the following specification:

- Resolution & Scale: ROIs were scanned at 40× magnification (0.22 µm/px) with a base resolution of 1024 × 1024 pixels.
- Contextual Data: Each sample includes a broader contextual ROI (5120 × 5120 pixels) centered around the primary area.
- Format: Annotations were provided in .GeoJSON format.

Scans are divided into either 'primary' or 'metastatic' and annotations were produced by a medical expert. Annotations were provided for both tissue and nuclei segmentation or classification.

1.1. Class Distribution and Imbalance

Investigation of the dataset revealed a significant class imbalance. We first inspect the training set for the tissue segmentation task, Table 1 shows pixel level statistics for each class. From this we can see that the tumor class appears most often, occurring in every image at least once and with a mean image coverage of 68.95%. Stroma and other occur less frequently. Further, we can see that the other class has a mean coverage of only 5.6% and a median of 1.27%, indicating that it often occurs in small regions of the image.

Additionally, from Figure 1 we can see that the tumor class has many images for which 100% of the pixels in the image are of the tumor class. Conversely, there are many images for which 0% of the pixels belong to the stroma or other class. The distribution for the other class is heavily skewed towards the left (0%) with a large spike of almost 100 images where no other class is present at all. Such a class imbalance is likely to impact the training performance of a segmentation model. This class imbalance informed the design choices detailed in later sections.

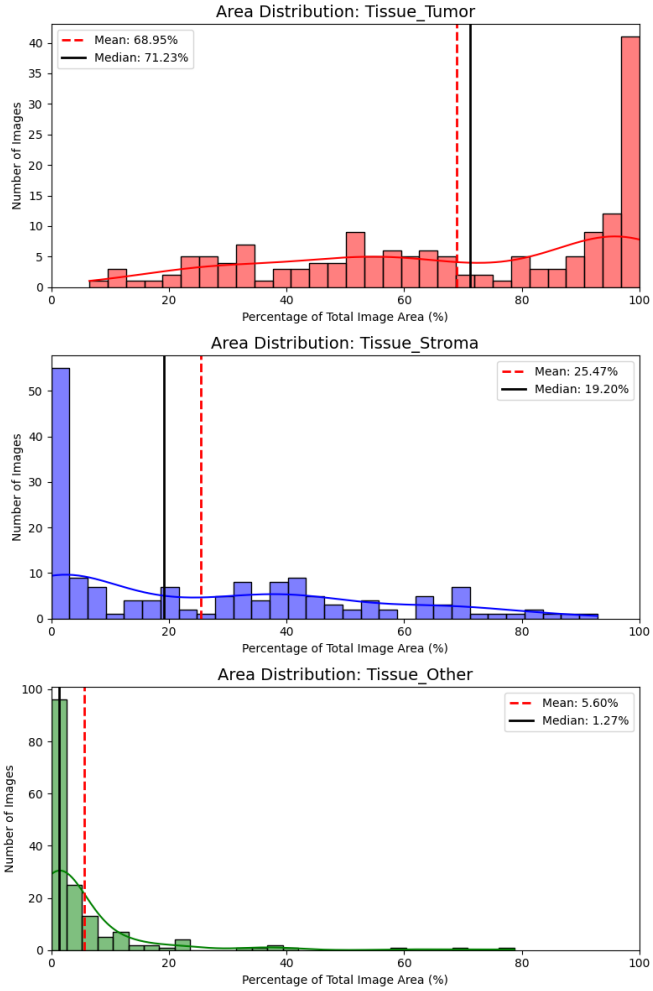


Figure 1. Distribution of each class within the train set as a proportion of the image.

2. Task 1: Tissue Segmentation

Tissue segmentation is a crucial task in computational histopathology. In this section, we formulate tissue segmentation as a three-class pixel-wise classification problem over the classes: Tissue Tumor, Tissue Stroma, and Other. As discussed in Section 1, our dataset is strongly imbalanced: the Tumor class dominates many images, while the Stroma and Other classes occupy smaller regions. This makes accurate segmentation more challenging. In this section, we compare two approaches: an end-to-end UNet trained directly for tissue segmentation, and a second model that uses an encoder pre-trained through image reconstruction, which is then adapted to the same downstream task. We then evaluate both methods quantitatively and qualitatively to determine whether pre-training improves segmentation performance under this class-imbalanced setting.

2.1. UNet-Based Segmentation

In this section, we implement an end-to-end segmentation network for the downstream tissue segmentation task. We use a UNet architecture to map histology images directly to pixel-wise labels over three classes: Tumor, Stroma, and Other, where Other includes blood vessel, epidermis, white background, and necrosis [7]. Tissue segmentation requires both global context and accurate localization. In histopathology images, Tumor and Stroma often occupy large portions of the slide, whereas classes such as Other tend to be rarer and more fragmented [5]. As a result, the model must preserve fine spatial detail while still reasoning over a wider tissue context. The encoder-decoder structure of UNet, together with skip connections, is well suited to this trade-off [7]. In addition, keeping the architecture relatively standard makes the comparison with the pre-trained model in Section 2.2 more fair. This is because any performance difference can be attributed more directly to pre-training rather than to a change in segmentation architecture. Our final UNet model contains 31.04M trainable parameters, which is substantially

Table 1. Coverage Statistics by Tissue Class

Class	Mean	Median	Max	Min	CoV	Images
tissue_tumor	68.95%	71.23%	100.00%	6.47%	0.409297	163
tissue_stroma	25.47%	19.20%	92.85%	0.00%	1.019260	115
tissue_other	5.60%	1.27%	78.72%	0.00%	2.080873	110

smaller than the baseline's $\sim$ 125M parameters.

2.1.1. Preprocessing

We first remap the GeoJSON annotations into the reduced label set $\{\text{Tumor}, \text{Stroma}, \text{Other}\}$. Grouping blood vessel, epidermis, white background, and necrosis into Other class reduces label sparsity and prevents the model from spending too much capacity on classes that are not the main focus of the task. We then obtain dense training masks by rasterizing the polygons onto the image grid. When multiple regions overlap, we draw the polygons in the order: Other, Stroma, and Tumor. This ordering strategy preserves the more diagnostically relevant tissue regions in the final mask. As such, less informative annotations do not overwrite Tumor or Stroma pixels at class boundaries.

We convert all images to floating point in $[0, 1]$ and normalize channel-wise using statistics computed on the training split only. Histopathology images can vary significantly in color and intensity due to staining and acquisition differences. Therefore, normalization helps stabilize the input distribution seen by the network. Let $x \in \mathbb{R}^{3\times H\times W}$ denote an input image. With $\mu = [0.6194,\ 0.4126,\ 0.6957]$, and $\sigma = [0.1986,\ 0.1939,\ 0.1411]$, the normalized input $\hat{x}$ is given by $\hat{x}_c = \dfrac{x_c - \mu_c}{\sigma_c}$. We apply this normalization consistently to the training, validation, and test sets in order to avoid a shift in input statistics between splits.

2.1.2. Network Design

Our segmentation network follows the standard UNet decomposition into an encoder $\mathcal{E}$, a bottleneck $\mathcal{B}$, and a decoder $\mathcal{D}$. Given an input image $\hat{x}$, the network produces segmentation logits $\ell = \mathcal{D}(\mathcal{B}(\mathcal{E}(\hat{x}))) \in \mathbb{R}^{3\times H\times W}$. Applying a softmax over the class dimension produces the following per-pixel class probabilities

$$p_c(u, v) = \frac{\exp(\ell_c(u, v))}{\sum_{k=1}^{3} \exp(\ell_k(u, v))}, \tag{1}$$

and the final prediction is

$$\hat{y}(u, v) = \arg \max_{c\in\{0,1,2\}} p_c(u, v). \tag{2}$$

The basic building block of our network is a double convolution unit f_{dc}. For a feature map h, we define it as

$$f_{dc}(h) = \psi(\text{GN}(\text{Conv}_{3\times3}(\psi(\text{GN}(\text{Conv}_{3\times3}(h)))))), \tag{3}$$

where GN denotes Group Normalization and ψ is a LeakyReLU activation with negative slope $\alpha = 0.1$. We chose Group Normalization over Batch Normalization because we performed training with a batch size of 4 [3, 16]. With such a small batch size, batch statistics can become noisy and make optimization unstable. On the other hand, Group Normalization is independent of the batch dimension and thus better suited to this setting [16].

Encoder The encoder contains four stages with channel progression $3 \to 64 \to 128 \to 256 \to 512$. This progressive increase in channel dimension allows the encoder to capture increasingly abstract tissue patterns as spatial resolution decreases. Let P denote 2×2 max-pooling. Then, the encoder features are

$$s_1 = f_{dc,1}(\hat{x}), \tag{4}$$
$$s_2 = f_{dc,2}(P(s_1)), \tag{5}$$
$$s_3 = f_{dc,3}(P(s_2)), \tag{6}$$
$$s_4 = f_{dc,4}(P(s_3)). \tag{7}$$

We retain the feature maps $\{s_1, s_2, s_3, s_4\}$ as skip connections for the decoder.

Bottleneck After the final pooling layer, the bottleneck expands the representation from 512 to 1024 channels: $z = \mathcal{B}(P(s_4)) = f_{dc,b}(P(s_4))$. This provides the model with the largest receptive field, which is useful for recognizing larger-scale tissue structures that can't be resolved from purely local features.

Decoder The decoder mirrors the encoder using transposed convolutions for upsampling. Let T_i denote a 2×2 transposed convolution, and let $[\cdot, \cdot]$ denote channel-wise concatenation. We then define the decoder by

$$h_1 = f_{dc,5}([T_1(z), s_4]), \tag{8}$$
$$h_2 = f_{dc,6}([T_2(h_1), s_3]), \tag{9}$$
$$h_3 = f_{dc,7}([T_3(h_2), s_2]), \tag{10}$$
$$h_4 = f_{dc,8}([T_4(h_3), s_1]). \tag{11}$$

The skip connections are important because they reintroduce the high-resolution spatial information lost during downsampling. This is especially useful in our setting, where tissue boundaries can be thin and irregular. We obtain the final logits with a 1×1 convolution applied to h_4. Table 2 summarizes the dimensionality of the feature maps throughout the network for a 512×512 training patch. It demonstrates that the spacial resolution halves at each encoder stage and doubles at each decoder block. Since the model is fully convolutional, the same architecture can also be applied at full image resolution during validation and testing.

2.1.3. Training Strategy

The main challenge in this task is not only the network architecture itself, but also how the model is exposed to the data during training. Although the original images are 1024×1024, we performed training on 512×512 patches. We chose this approach for two reasons. Firstly, it reduces memory usage and makes training with a deeper UNet feasible at batch size 4. Secondly,

Table 2. Tensor dimensionality throughout the UNet architecture for a 512×512 input patch.

Stage	**Operation**	**Output** ($C \times H \times W$)
Input	Input patch $\hat{x}$	$3 \times 512 \times 512$
Encoder 1	f_{dc}	$64 \times 512 \times 512$
Encoder 2	$P \to f_{dc}$	$128 \times 256 \times 256$
Encoder 3	$P \to f_{dc}$	$256 \times 128 \times 128$
Encoder 4	$P \to f_{dc}$	$512 \times 64 \times 64$
Bottleneck	$P \to f_{dc}$	$1024 \times 32 \times 32$
Decoder 1	$T + s_4 \to f_{dc}$	$512 \times 64 \times 64$
Decoder 2	$T + s_3 \to f_{dc}$	$256 \times 128 \times 128$
Decoder 3	$T + s_2 \to f_{dc}$	$128 \times 256 \times 256$
Decoder 4	$T + s_1 \to f_{dc}$	$64 \times 512 \times 512$
Output	1×1 Conv	$3 \times 512 \times 512$

it improves the quality of the supervision signal. Many full-resolution images contain large homogeneous regions, particularly from the dominant Tumor class. Some images also contain large background-like regions that are grouped into Other. If training is performed only on such images, the model can bias toward learning to focus on these easier regions. At the same time, smaller or more difficult structures like Stroma and tissue boundaries would have less influence on the loss. Our approach helps address this issue by providing the network more frequently with informative local regions.

To improve coverage of under-represented tissue types, we constructed each training epoch from 2400 class-aware sampled patches. We define the sampling distribution by $p_{\text{tumor}} = 0.37$, $p_{\text{stroma}} = 0.37$, and $p_{\text{other}} = 0.10$, where the remaining probability is assigned to random sampling. We assign equal probability to Tumor and Stroma because Stroma is under-represented in terms of pixels but remains clinically important. Therefore, it should appear more frequently during training than naive random sampling would provide. We also use ramping to gradually increase the probability of drawing on Other-centered patch during the first 10 epochs. This prevents the early stages of training from being dominated by the highly heterogeneous Other class. It further allows the network to first learn stronger decision boundaries for Tumor and Stroma classes.

In addition, a sampled patch is accepted only if it contains a sufficient proportion of the target class and a sufficient amount of labeled tissue content. We implemented it to avoid wasting training updates on patches that are visually uninformative. In practice, this makes the training batches more balanced and was one of the main reasons patch-based training performed better than full-image training.

To improve robustness, we apply data augmentation during training using random horizontal flips, random vertical flips, random 90° rotations, and mild color jitter. We kept these augmentations intentionally simple. Histopathology images can appear under different orientations, so we expect mild variation in appearance. However, we don't perform aggressive transformations because it introduces the risk of drastically changing morphology. We therefore use augmentations that improve invariance while still preserving tissue structure. We apply the same geometric transformation jointly to the image and its corresponding mask to preserve pixel alignment.

2.1.4. Loss Function and Optimization

We trained the model using the following hybrid objective function: $\mathcal{L}_{seg} = \lambda_{CE}\mathcal{L}_{CE} + \lambda_{Dice}\mathcal{L}_{Dice}$, with $\lambda_{CE} = \lambda_{Dice} = 0.5$. We chose this function because cross-entropy provides stable pixel-wise supervision, and Dice directly optimizes region overlap. This is useful in class-imbalanced segmentation problems because a model can obtain reasonable cross-entropy by performing well on the dominant class, while still giving poor overlap on smaller regions [10].

Weighted Cross-Entropy To compensate for class imbalance, we derive the class weights from the training-set pixel frequencies using an inverse square-root rule $w_c = \dfrac{1}{\sqrt{f_c}}$, and then normalize relative to the smallest weight. We use the inverse square-root form rather than full inverse frequency weighting because it increases the contribution of rare classes without making the loss excessively dominated by them. After scaling and clipping, we used the following the final class weights during training

$$[\tilde{w}_{\text{tumor}},\ \tilde{w}_{\text{stroma}},\ \tilde{w}_{\text{other}}] = [1.0000,\ 1.6489,\ 3.0000]. \quad (12)$$

The relatively large weight assigned to Other reflects the fact that it is a visually heterogeneous class, making it difficult to model consistently. We also use label smoothing with $\epsilon = 0.02$ [11], so that the softened target distribution is

$$y'_c = (1 - \epsilon)y_c + \frac{\epsilon}{3}, \quad (13)$$

and the weighted cross-entropy term is

$$\mathcal{L}_{CE} = -\frac{1}{HW}\sum_{u=1}^{H}\sum_{v=1}^{W}\sum_{c=1}^{3}\tilde{w}_c\, y'_c(u,v)\log p_c(u,v). \quad (14)$$

We do so to reduce over-confident predictions at ambiguous tissue boundaries, where the rasterized labels may not be perfect.

Dice Loss We computed the Dice term from the softmax probabilities. For class c, we have

$$\text{Dice}_c = \frac{2\sum p_c y_c + \varepsilon}{\sum p_c + \sum y_c + \varepsilon}, \quad (15)$$

and the corresponding loss is

$$\mathcal{L}_{Dice} = 1 - \frac{1}{3}\sum_{c=1}^{3}\text{Dice}_c. \quad (16)$$

This term directly encourages overlap between predicted and ground-truth regions, which is especially useful when minority classes occupy small areas.

Optimization We performed the training for 50 epochs using AdamW with learning rate 3×10^{-4}, weight decay 10^{-4}, and batch size 4 [6]. We chose AdamW because it gives stable optimization for deep convolutional networks. We also used the

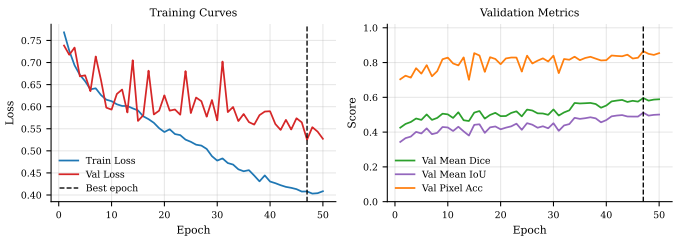


Figure 2. Training and validation curves for the UNet model, including loss and validation-set segmentation metrics used for checkpoint selection.

following cosine annealing schedule to decay the learning rate [6]:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min})\left(1 + \cos\left(\frac{\pi t}{T_{\max}}\right)\right), \quad (17)$$

where $T_{\max} = 50$ and $\eta_{\min} = 3 \times 10^{-5}$. This allows the learning rate to decrease smoothly over training rather than dropping sharply. During training, we also use automatic mixed precision and gradient clipping with $\|g\|_2 \leq 1.0$. We do so to improve numerical stability under the memory constraints of the training setup. The model checkpoint used for final evaluation is selected according to the validation mean Dice score. Figure 2 shows the training and validation curves for this model.

In summary, our key design choices were: (1) Class-aware patch-based sampling to handle class imbalance and ease computational constraints; (2) Hybrid DICE + cross-entropy loss to balance region and pixel-level accuracy; (3) class-weighted loss to increase the contribution of rare classes to the network loss.

2.2. Autoencoder Pre-Training for Segmentation

The objective of pre-training is to utilise unlabelled data in order to obtain model weights which are more suited for the downstream task. In this case, we pre-trained with the task of noisy image reconstruction of the training data with the intent for the network encoder to learn relevant feature representations of the data. This pre-trained encoder would then be used as the feature extractor for the downstream segmentation task.

2.2.1. Network Design

In order to isolate the effect of pre-training on segmentation performance, the segmentation network architecture is identical to the network described in Section 2.1. The primary difference for this approach is that the encoder is frozen (i.e. the weights are not updated during training) and the encoder weights are initialised using the weights produced from the pre-training step.

The image reconstruction autoencoder is similar to the segmentation network with some adaptations to support the task of noisy image reconstruction. The image reconstruction network follows a symmetric convolutional autoencoder structure, consisting of an encoder $\mathcal{E}$, a bottleneck $\mathcal{B}$, and a decoder $\mathcal{D}$. The total transformation for a corrupted input image $\tilde{x} \in \mathbb{R}^{3\times H\times W}$ is defined as $\hat{x} = \sigma(\mathcal{D}(\mathcal{B}(\mathcal{E}(\tilde{x}))))$.

Crucially, to force the network to learn a compressed global representation through the bottleneck z, skip connections are omitted.

2.2.2. Noisy Image Reconstruction

The pre-training task of noisy image reconstruction with autoencoders was introduced by Vincent et al. [14], who demonstrate that it allows for the learned representations to better capture the structure of the input distribution, leading to intermediate representations better suited for subsequent learning tasks.

The network is passed a corrupted image at train time, and is evaluated against that same image prior to corruption. The objective is to pass the corrupted image through the encoder-decoder network to produce the uncorrupted image. In order to implement this, the following noise augmentations were applied stochastically at train time. Let $x \in \mathbb{R}^{3\times H\times W}$ be the original image patch. The corrupted input $\tilde{x}$ is generated via a transformation function $\mathcal{C}(x)$, defined as:

$$\tilde{x} = \mathcal{G}_\sigma(\mathcal{J}(x)) \quad (18)$$

where:

- $\mathcal{J}(\cdot)$ denotes a color jittering operation. For each pixel intensity p, the brightness and contrast are adjusted by a random factor $f \in [1-\epsilon, 1+\epsilon]$, with $\epsilon = 0.2$.
- $\mathcal{G}_\sigma(\cdot)$ represents a Gaussian blur kernel of size $k \times k$ ($k = 5$). $\sigma \sim \mathcal{U}(0.1, 1.0)$.

By training the autoencoder to minimize the reconstruction error between x and $\mathcal{D}(\mathcal{E}(\tilde{x}))$, the encoder is forced to "denoise" the input, effectively learning to ignore local pixel-level perturbations and focus on the underlying global structure and histological features of the tissue.

2.2.3. Training

In order to train the autoencoder, the same patch-based sampling approach as described in Section 2.1 was used. This ensures that the encoder network was trained under the same image conditions which the downstream segmentation task expects, which are designed to minimise the impact of class imbalance. The autoencoder was trained for 50 epochs using the Adam optimizer with a learning rate of 3×10^{-4} and a batch size of 4. The autoencoder has around 31.04M trainable parameters.

Loss Function To ensure that the reconstructed images $\hat{x}$ preserved both pixel-level accuracy and structural consistency with the original clean patches x, a hybrid loss function $\mathcal{L}_{total}$ was employed. This loss combines Mean Squared Error (MSE) and the Structural Similarity Index Measure (SSIM) [15]:

$$\mathcal{L}_{total} = \lambda \cdot \mathcal{L}_{MSE}(x, \hat{x}) + (1 - \lambda) \cdot (1 - \text{SSIM}(x, \hat{x})) \quad (19)$$

where λ was set to 0.5 to provide equal weighting to both terms. The MSE term, defined as $\frac{1}{N}\sum(x_i - \hat{x}_i)^2$, enforces intensity fidelity, while the SSIM term ensures that the encoder captures higher-level structural patterns such as nuclei boundaries and tissue textures.

Mathematically, $\text{SSIM}(x, \hat{x})$ is defined as the product of three comparison measurements: luminance $l(x, \hat{x})$, contrast $c(x, \hat{x})$, and structure $s(x, \hat{x})$:

$$\text{SSIM}(x, \hat{x}) = [l(x, \hat{x})]^\alpha \cdot [c(x, \hat{x})]^\beta \cdot [s(x, \hat{x})]^\gamma \quad (20)$$

With the exponents typically set to $\alpha = \beta = \gamma = 1$, the specific components are calculated using the means $(\mu_x, \mu_{\hat{x}})$,

variances ($\sigma_x^2, \sigma_{\hat{x}}^2$), and covariance ($\sigma_{x\hat{x}}$) of the patches:
Luminance: $l(x, \hat{x}) = \frac{2\mu_x\mu_{\hat{x}}+C_1}{\mu_x^2+\mu_{\hat{x}}^2+C_1}$
Contrast: $c(x, \hat{x}) = \frac{2\sigma_x\sigma_{\hat{x}}+C_2}{\sigma_x^2+\sigma_{\hat{x}}^2+C_2}$
Structure: $s(x, \hat{x}) = \frac{\sigma_{x\hat{x}}+C_3}{\sigma_x\sigma_{\hat{x}}+C_3}$
where C_1, C_2, and C_3 are small constants added to stabilize the division when the denominators are close to zero. By optimizing for $(1 - \text{SSIM})$, the model is specifically penalized for distorting nuclei boundaries and tissue textures, which are critical for downstream analysis.

Downstream Segmentation Following the pre-training phase, the weights of the encoder $\mathcal{E}$ were extracted and loaded into the segmentation UNet. To strictly evaluate the feature extraction capabilities of the autoencoder, the encoder weights were frozen. Training of the segmentation network followed the same setup as described in Section 2.1 including inverse frequency weighting of classes within the loss function in order to minimise the impact of the class imbalance.

An additional experiment was conducted with the encoder parameters unfrozen (trainable) with a low learning rate (10^{-5}) to determine if adjusting the feature representation specifically for the segmentation task would improve performance.

2.3. Evaluation and Comparison

2.3.1. Metrics

We evaluate segmentation quality using Dice, IoU, and pixel accuracy. For class c, let P_c and G_c denote the predicted and ground-truth pixel sets. We define

$$\text{Dice}_c = \frac{2|P_c \cap G_c|}{|P_c| + |G_c|}, \qquad \text{IoU}_c = \frac{|P_c \cap G_c|}{|P_c \cup G_c|}. \tag{21}$$

We report mean Dice and mean IoU by averaging over the three classes. We use mean Dice as the main metric for checkpoint-selection because it gives equal importance to each class despite the strong pixel imbalance in the dataset. In addition, we report pixel-level precision, recall, and F1-score in order to better understand the class-specific prediction behavior.

2.3.2. End-to-End UNet

We first evaluate the end-to-end UNet on the validation and test sets. During training, the validation mean Dice improved steadily and reached its best value of 0.5959 at epoch 47. Using this checkpoint, the model obtained a validation loss of 0.5253, a mean IoU of 0.5117, and a pixel accuracy of 0.8651. On the test set, the model achieved a loss of 0.6086, a mean Dice of 0.5851, a mean IoU of 0.4916, and a pixel accuracy of 0.8332. Since the drop in mean Dice from validation to test is only 0.0108, we do not observe a large generalization gap. This suggests that the patch-based sampling strategy, class-aware loss weighting, and moderate augmentation were sufficient to limit overfitting.

At the class level, the model performs best on Tumor. On the test set, Tumor achieves Dice 0.8524, IoU 0.7937, precision 0.9028, recall 0.8950, and F1-score 0.8988. We found Stroma more challenging, but our model still scores Dice 0.6142, IoU 0.4842, precision 0.6223, recall 0.6886, and F1-score 0.6538. Our most difficult class was the Other class, with Dice 0.2885, IoU 0.1968, precision 0.5376, recall 0.4121, and F1-score 0.4665. We also observed a similar pattern on the validation set, where Tumor achieves Dice 0.8628, Stroma achieves Dice 0.6516, but Other remains much lower at 0.2733. We believe this is because Other occupies smaller and more heterogeneous regions than Tumor and Stroma. This makes it difficult to segment consistently.

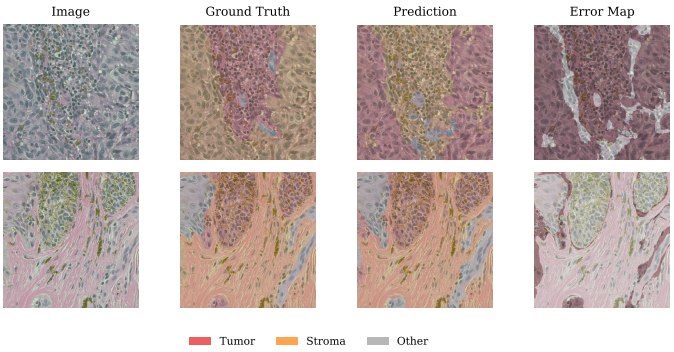


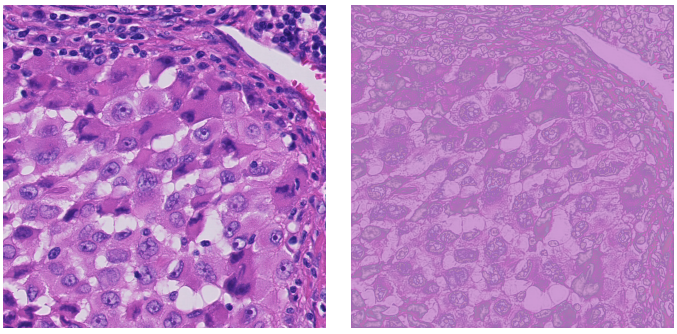
Figure 3. Qualitative segmentation results for the end-to-end UNet on the test set. The top row shows the worst example by mean Dice, while the bottom row shows the best example.

Overall, these results indicate that the end-to-end UNet learns a useful representation for tissue segmentation and produces strong predictions for Tumor and Stroma. However, the quantitative and qualitative results also show that small or ambiguous Other regions remain the main source of error. Moreover, in Figure 3, we observe that the worst-case example contains larger boundary and minority-region errors than the best-case example.

2.3.3. Autoencoder Pre-Training for Segmentation

Here we evaluate the performance of both the image reconstruction autoencoder and the downstream segmentation model using the learned encoder.

The de-noising image reconstruction autoencoder achieved a final validation loss of 0.596 after 10 epochs using the hybrid L1 and SSIM loss. Figure 4 shows an example of the produced reconstructions for the validation set (no noise augmentation).



(a) Original Input (Validation) (b) Autoencoder Reconstruction

Figure 4. Output of the Denoising Autoencoder on a validation image.

We experimented with two different approaches of applying this learned encoder to the downstream segmentation task. The first was to keep the encoder entirely frozen, using it only as a feature extractor, the second was to allow the encoder parame-

ters to be trainable with a lower learning rate (1e-5) in order to allow it to adjust to the specific requirements of the task, while not entirely forgetting the learned representations.

The segmentation model trained with the frozen encoder obtained a final average DICE score of 0.5545 on the test set, while the model trained with the unfrozen encoder obtained a final average DICE score of 0.5946. From this we can determine that while the feature representations learned through the image reconstruction pre-training are suitable for downstream segmentation, fine-tuned adjustment of these weights produces a more powerful model. Table 3 summarises the segmentation performance of each model.

Table 3. Segmentation performance and parameter efficiency.

Model Strategy	Dice	Δ Dice	Params
Baseline	0.4670	—	125.0 M
End-to-End UNet	0.5851	+0.1181	31.0M
Pre-Trained (Frozen)	0.5545	+0.0875	26.3 M
Pre-Trained (Unfrozen)	0.5946	+0.1276	31.0 M

2.3.4. Comparison

Table 3 summarises the test-set Dice scores for all three approaches alongside the provided baseline. All three of our models substantially outperform the baseline (0.4670).

The end-to-end UNet achieves a test Dice of 0.5851, a $+0.1181$ improvement over the baseline despite using significantly fewer parameters (31.0M vs 125M), demonstrating the success of our class-aware patch sampling, hybrid loss weighting, and data augmentation. The frozen pre-trained model scores 0.5545, falling 0.0306 below the end-to-end UNet. While the autoencoder does learn meaningful feature representations through noisy image reconstruction, these are not sufficiently specialised for segmentation when the encoder is kept fixed, as the reconstruction task does not optimise for the features most beneficial to this downstream objective.

Allowing the encoder to fine-tune with a reduced learning rate of 10^{-5} recovers this gap and yields the strongest overall result at 0.5946, surpassing the end-to-end UNet by 0.0095. The low learning rate allows the encoder to adapt specifically for segmentation while preserving the general feature understanding acquired during pre-training. However, the margin over the end-to-end UNet is notably small, which can be attributed to the effectiveness of the patch-based training strategy in Section 2.1. By constructing class-aware batches and filtering uninformative patches, the end-to-end UNet is already exposed to a diverse and balanced distribution of tissue types, reducing the relative advantage that pre-training would otherwise offer.

In summary, autoencoder pre-training provides the most value when the encoder is fine-tuned to the downstream task, but gains over a well-trained end-to-end model are limited when labelled data is used effectively.

3. Task 2: Nuclei Classification

In this section, we focus on nuclei classification, which complements tissue segmentation by operating at the cellular rather than the tissue level. We classify image patches into three nuclei classes: Tumor, Lymphocyte, and Histiocyte. This task is relatively more local in nature because each input contains only a small neighborhood around a single nucleus. As a result, the model cannot rely on broad tissue context and must instead distinguish classes using relatively subtle morphological and appearance cues. For this reason, both dataset creation and model design are important for this task. We first construct balanced patch-level datasets from the nuclei annotations and then train an end-to-end classification network. We later compare this supervised model with contrastive pre-training in order to determine whether representation learning improves downstream nuclei classification.

3.1. Dataset Creation

3.1.1. Patch Extraction

We create patch-level datasets directly from the nuclei annotations in the training and validation splits. We only retain the three nuclei classes relevant to this task: Tumor, Lymphocyte, and Histiocyte. For each annotated nucleus polygon g, we compute its centroid $c(g) = (c_x, c_y)$, and then extract a square RGB patch centered at that location. Let I denote the original image. We then define the extracted patch as $x = \mathcal{P}(I, c(g)) \in \mathbb{R}^{3\times100\times100}$, where $\mathcal{P}$ denotes centroid-centered patch extraction. This uses zero-padding whenever the crop extends beyond the image boundary.

We choose this construction for two reasons. Firstly, the provided test set consists of 100×100 nucleus-centered patches, so our construction keeps the training and validation data in the same format as the test set. It reduces the risk of mismatch between training and evaluation. Secondly, our construction ensures that the model sees the target cell together with a small amount of surrounding context since each patch is centered on the annotated nucleus. This is useful because some classes share similar uclear appearance, and a small amount of local neighborhood information can help distinguish them without introducing too much irrelevant background.

3.1.2. Balanced Sampling Strategy

If we had sampled uniformly from all available nuclei, there would be a possibility of having a small number of densely annotated regions of interest dominating the dataset. We want to avoid this because the classifier could then learn slide-specific appearance or staining cues rather than class-specific nuclear morphology. We therefore group candidates by ROI identifier inferred from the filename and, when possible, also by source type such as primary or metastatic. We then sample evenly across these groups so that a small number of ROIs do not dominate any class.

Let $\mathcal{C}_c^{\text{train}}$ and $\mathcal{C}_c^{\text{val}}$ denote the candidate pools for class $c \in \{\text{tumor}, \text{lymphocyte}, \text{histiocyte}\}$ in the training and validation splits. From these pools, we construct the supervised datasets

$$\mathcal{D}_c^{\text{train}} \subset \mathcal{C}_c^{\text{train}}, \qquad |\mathcal{D}_c^{\text{train}}| = 2500, \tag{22}$$

$$\mathcal{D}_c^{\text{val}} \subset \mathcal{C}_c^{\text{val}}, \qquad |\mathcal{D}_c^{\text{val}}| = 700. \tag{23}$$

This gives us a balanced training set with 7500 patches in total and a balanced validation set with 2100 patches.

We also create a contrastive set for the experiments in Section 3.3. For each class, we sample $|\mathcal{D}_c^{\text{contr}}| = 1000$ examples

from the training split. However, we explicitly exclude any coordinate already used in the supervised training set. In other words, for every class c, we satisfy $\mathcal{D}_c^{\text{train}} \cap \mathcal{D}_c^{\text{contr}} = \emptyset$. This ensures that there is no learkage between the supervised and contrastive datasets. We apply the same ROI-balanced strategy to the validation split to ensure that it reflects a broader range of nuclei appearances rather than only a small subset of regions. We keep the provided test set completely separate and use it only for final evaluation.

3.2. End-to-End Classification Network

The objective of this part is to train an end-to-end classifier for the downstream nuclei classification task. We use an EfficientNet-B0 architecture to map each nuclei patch to one of three labels: Tumor, Lymphocyte, or Histiocyte [12]. We choose EfficientNet-B0 because our dataset is relatively small. We therefore want a model that can learn discriminative nuclei features without being unnecessarily large. In practice, this architecture gives us a good balance between representation capacity and generalization. Moreover, it contains 4.34M trainable parameters and thus stays close to the scale of the baseline, which has ~ 5.0M parameters.

We opted to initialise the model weights using an ImageNet pre trained model, in order to avoid the initial learning of generic features. We also trained EfficientNet-B0 from scratch as a comparison. The initial from-scratch model achieved a test accuracy of 69.53%. We then improved this setting by using dataset-specific normalization based on the training-set RGB statistics, and by extending the augmentation pipeline with color jitter and random autocontrast in addition to the geometric flips and rotations. These two improvements increased the test accuracy to 74.27%. However, this was still below the 77.93% achieved by our final ImageNet pretrained model. We note that our final ImageNet pretrained model did not use these two additional changes. While this is not a strict controlled ablation, it still suggests that pretrained initialization gave a stronger starting point for our relatively small nuclei classification dataset. For this reason, we initialize the final model with ImageNet weights rather than training it from scratch.

3.2.1. Network Design

Let $x \in \mathbb{R}^{3\times 100\times 100}$ denote an extracted nuclei patch. Before classification, we apply an input transform $\mathcal{T}$ that resizes the patch to 224×224 and normalizes it using the ImageNet statistics. This gives a transformed input

$$\tilde{x} = \mathcal{T}(x) \in \mathbb{R}^{3\times 224\times 224}. \tag{24}$$

We then pass $\tilde{x}$ through an EfficientNet-B0 encoder $\mathcal{E}$. The encoder produces a feature representation $h \in \mathbb{R}^{1280}$ after global pooling, and our custom classification head $\mathcal{H}$ maps this representation to logits

$$\ell = \mathcal{H}(\mathcal{E}(\tilde{x})) \in \mathbb{R}^3. \tag{25}$$

We obtain the final class probabilities through

$$p_c = \frac{\exp(\ell_c)}{\sum_{k=1}^{3} \exp(\ell_k)}, \tag{26}$$

and we predict the class label by

$$\hat{y} = \arg \max_{c\in\{0,1,2\}} p_c. \tag{27}$$

Rather than keeping the default EfficientNet classifier, we replace it with a lightweight task-specific head $\mathcal{H}$. Let ϕ denote the SiLU activation and D_r denote dropout with rate r. We then define the classifier head as

$$\mathcal{H}(h) = W_2\Big(D_{0.2}\big(\phi(W_1(D_{0.3}(h)))\big)\Big) + b_2, \tag{28}$$

where $W_1 : \mathbb{R}^{1280} \to \mathbb{R}^{256}$ and $W_2 : \mathbb{R}^{256} \to \mathbb{R}^3$. This customization ensures that the classifier can better adapt the pretrained EfficientNet features to our nuclei classification task. We use dropout to regularize the classifier and reduce overfitting [9].

Unlike the contrastive setting in Section 3.3, we do not freeze the encoder here. We keep the full network trainable because the purpose of this end-to-end model is to learn task-specific features directly from the labeled nuclei patches. This is particularly important because nuclei morphology is significantly different from the natural-image patterns on which the pretrained weights were originally learned.

3.2.2. Input Processing and Augmentation

We use ImageNet normalization with mean $\mu = [0.485,\ 0.456,\ 0.406]$ and standard deviation $\sigma = [0.229,\ 0.224,\ 0.225]$ so that the input distribution remains consistent with the pretrained EfficientNet encoder [2]. For the evaluation pipeline, we therefore use

$$\mathcal{T}_{\text{eval}}(x) = \frac{\mathcal{R}(x) - \mu}{\sigma}, \tag{29}$$

where $\mathcal{R}$ denotes the resizing of the input patch to 224×224 followed by conversion to floating point. For training, we use an augmented transform $\mathcal{T}_{\text{train}}$ that adds random horizontal flips, random vertical flips, random 90° rotations, and color jitter before normalization.

We choose these augmentations because nuclei do not have a fixed orientation, and we expect mild color variation in histopathology images due to staining differences. At the same time, we avoid overly aggressive transformations because they could distort the fine morphological cues that distinguish the three nuclei classes. Figure 5 shows representative examples of the training-time augmentations.

We also explicitly remap the class indices to the fixed order $\{\text{tumor},\ \text{lymphocyte},\ \text{histiocyte}\}$. This is necessary because the class order inferred from folders may not match the label order used elsewhere in our pipeline, especially for the separately provided test set.

3.2.3. Training

We train the classifier using cross-entropy loss with label smoothing [11]. For a one-hot target y and smoothing factor $\epsilon = 0.05$, we define the softened target distribution as in Equation (13). We then optimize

$$\mathcal{L}_{cls} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{c=1}^{3} y'_{i,c} \log p_{i,c}, \tag{30}$$

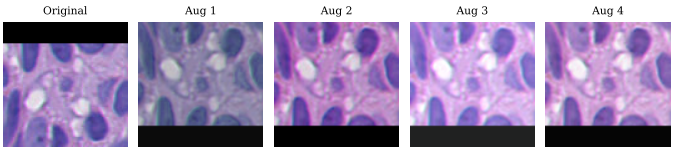


Figure 5. Representative examples of the training-time augmentations used for nuclei classification.

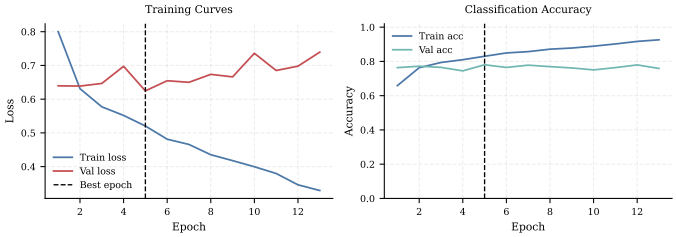


Figure 6. Training and validation curves for the end-to-end EfficientNet-B0 classifier.

where B denotes the minibatch size. We use label smoothing because some nuclei can be visually ambiguous even when the annotation is correct. Therefore, encouraging slightly less confident predictions helps the model generalize better.

We optimize the network with AdamW using learning rate 3×10^{-4}, weight decay 10^{-4}, and batch size 32 [6]. We also use the same cosine annealing schedule used in Equation (17), with $T_{\max} = 50$ and $\eta_{\min} = 0.05\,\eta_{\max}$. We choose AdamW because it provides stable fine-tuning behavior for pretrained convolutional networks. In addition, we use cosine annealing because it decreases the learning rate smoothly over training rather than introducing sharp drops. Likewise, we use early stopping based on validation accuracy with patience 7 and minimum improvement threshold 0.001. We use early stopping because the dataset is limited in size, and longer training can quickly lead to overfitting even when we use data augmentation. We therefore select the best checkpoint according to validation accuracy rather than the final training epoch. Figure 6 shows the training and validation curves for this model.

3.3. Contrastive Learning Pre-Training for Classification

The objective of contrastive pre-training is to learn an encoder $\mathcal{E}$ that maps images into a latent space where semantically similar samples are clustered together while dissimilar ones are separated. We explore two strategies: a purely unsupervised approach following the SimCLR framework [1] and a Supervised Contrastive (SupCon) approach [4].

3.3.1. Network Design

The classification architecture matches that of the end-to-end classification task described in Section 3.2, i.e. an EfficientNet-B0 network with 4.1M parameters. However, the encoder weights are pre-trained on the contrastive learning task, and remain frozen. During the pre-training phase, the network is augmented with a projection head $\mathcal{P}$.

- **Encoder** $\mathcal{E}$: Maps an input image $\tilde{x} \in \mathbb{R}^{3\times224\times224}$ to a feature vector $h \in \mathbb{R}^{1280}$ via Global Average Pooling.
- **Projection Head** $\mathcal{P}$: A Multi-Layer Perceptron (MLP) consisting of a linear layer ($1280 \to 512$), Batch Normalization, ReLU activation, and a final linear layer ($512 \to 128$).

The final output $z = \mathcal{P}(\mathcal{E}(\tilde{x}))$ is L_2-normalized such that $\|z\|_2 = 1$, ensuring that the contrastive loss is calculated based on cosine similarity on a unit hypersphere.

3.3.2. Contrastive Loss Functions

For each input x, two stochastic augmentations $t, t' \sim \mathcal{T}$ are applied to produce two correlated views, $\tilde{x}_i$ and $\tilde{x}_j$. These views are processed by an encoder $\mathcal{E}$ (EfficientNet-B0) and a projection head $\mathcal{P}$ to produce embeddings $z = \mathcal{P}(\mathcal{E}(\tilde{x}))$.

Unsupervised Contrastive Learning (SimCLR) In the unsupervised setting, the model employs the NT-Xent (Normalized Temperature-scaled Cross Entropy) loss. For a positive pair of augmented views (i, j) derived from the same image, the loss is:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{I}_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau)} \tag{31}$$

where $\text{sim}(\cdot)$ denotes cosine similarity and τ is a temperature hyperparameter. This forces the model to learn instance-specific features without the need for labels.

Supervised Contrastive Learning (SupCon) The SupCon objective leverages class labels to pull together all samples k in a batch belonging to the same class $y_i = y_k$:

$$\mathcal{L}_i^{sup} = \sum_{i\in I} \frac{-1}{|P(i)|} \sum_{p\in P(i)} \log \frac{\exp(\text{sim}(z_i, z_p)/\tau)}{\sum_{a\in A(i)} \exp(\text{sim}(z_i, z_a)/\tau)} \tag{32}$$

where $P(i)$ is the set of indices of all samples in the batch sharing the same label as i. This encourages the latent space to be discriminative of the underlying tissue types (Tumor, Lymphocyte, Histiocyte).

3.3.3. Training

The pre-training was conducted for 60 epochs using the **Adam optimizer** with a learning rate of 3×10^{-4} and a batch size of 64.

Multi-View Augmentations A stochastic augmentation pipeline $\mathcal{T}$ generated two views for every image. Transformations included : **Geometric**: Random Resized Crops (224×224), Horizontal/Vertical Flips, and $180°$ Rotations. **Noisy**: Color Jittering (brightness, contrast, and saturation adjusted by a factor of 0.2) and Gaussian Blurring ($k = 3$).

Downstream Classification Following the pre-training, the projection head $\mathcal{P}$ was discarded and the encoder $\mathcal{E}$ was frozen. A linear classification head ($1280 \to 3$) was then trained on the fixed features.

By freezing the EfficientNet-B0 encoder, the number of trainable parameters during the downstream task was reduced from around 4.1M to only 3,843 (the linear head). This represents a significant reduction in computational complexity compared to the 5M parameter baseline.

3.4. Evaluation and Comparison

3.4.1. End-to-End Classification Network

We first evaluate the end-to-end EfficientNet-B0 classifier on the validation and test sets. During training, the model reached its best validation accuracy of 78.95% at epoch 5. After this epoch, the validation performance fluctuated while the training accuracy continued to increase. We therefore used early stopping and selected the checkpoint from epoch 5 for final evaluation. This behavior suggests that the model learned discriminative features relatively quickly, but also that continuing to optimize beyond this point mainly improved fit to the training data rather than generalization.

Using the selected checkpoint, we obtained a validation loss of 0.6252 and a validation accuracy of 78.95%. On the test set, the model achieved a loss of 0.6264 and an accuracy of 77.93%. Since the test accuracy is only slightly lower than the validation accuracy, we do not observe a meaningful generalization drop between the two splits. This indicates that the ROI-balanced dataset construction and the augmentation strategy were sufficient to limit overfitting. This is particularly important given the relatively small size of our nuclei classification dataset.

At the class level, the model performs the best on Tumor, while Lymphocyte also performs reasonably well. On the test set, Tumor achieves precision 0.8471, recall 0.7914, and F1-score 0.8183, while Lymphocyte achieves precision 0.8011, recall 0.8000, and F1-score 0.8006. On the other hand, Histiocyte remains the most difficult class, with precision 0.6614, recall 0.7293, and F1-score 0.6937. We believe this is because Histiocyte shows greater visual overlap with the other nuclei classes and is represented by fewer test samples than Tumor and Lymphocyte. We also observe a similar pattern on the validation set, where Lymphocyte achieves the highest recall (0.8986) and Histiocyte obtains the lowest recall (0.7086).

We also evaluated performance separately for primary and metastatic test samples using the checkpoint selected by early stopping. At epoch 5, our model achieved an accuracy of 77.15% on primary samples and 79.17% on metastatic samples. Although the metastatic subset gives slightly higher accuracy, the macro F1-score is slightly higher on the primary subset (0.7593 compared to 0.7362). We believe that this difference is fairly small and is likely explained at least partly by the class composition of the two subsets, rather than by source type alone. In addition, the metastatic subset constains more Tumor samples, and Tumor is the class on which our model performs the best. We also observed that when training was continued for longer, the overall test accuracy decreased and the gap between primary and metastatic performance became larger. This suggests that the later model was overfitting rather than learning a more meaningful difference between the two source types. We therefore find only limited evidence that the source of the sample (primary or metastatic) has a large independent effect on performance.

We also evaluate the model qualitatively using representative test predictions. Figure 7 shows both confident correct predictions and high-confidence mistakes. We observe that the correct predictions usually correspond to nuclei with clearer morphology. In contrast, the mistakes tend to occur for more visually ambiguous nuclei, especially when Histiocyte overlaps in appearance with the other two classes. This is consistent with the class-wise results above and helps explain why Histiocyte remains the most difficult class.

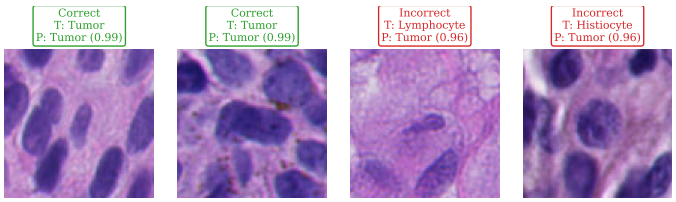


Figure 7. Qualitative evaluation of the end-to-end EfficientNet-B0 classifier on the test set, showing representative correct and incorrect predictions.

Overall, these results indicate that the end-to-end classifier learns a useful representation for nuclei classification and performs above the baseline accuracy of 70.83%. This is notable because our model uses only 4.34M trainable parameters, which is slightly smaller than the approximate 5M baseline. Our results suggest that careful dataset construction, moderate augmentation, and a lightweight pretrained architecture are already sufficient to obtain strong downstream classification performance.

3.4.2. Contrastive Pre-Training

Evaluation of Contrastive Pre-Training In order to verify the success of each contrastive learning method prior to the downstream classification task we employed metrics such as per-class intra- and inter-cosine similarity (describing the similarity to same-class and different-class instances, respectively) as well as t-SNE [13] visualisations of the embedding into 2 dimensions.

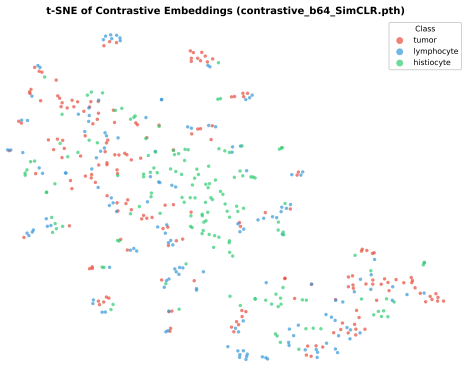
t-SNE is a non-linear dimensionality reduction technique that maps the 1280-dimensional feature vectors $h \in \mathbb{R}^{1280}$ onto a 2D plane while preserving local neighbourhood structures allowing us to visually inspect whether the quality of the contrastive learning.

Figure 8 visualises how well each method grouped data points together. We can see that the supervised contrastive learning approach generates more distinct clusters than the unsupervised approach. However, neither approach achieves a perfect separation of classes.

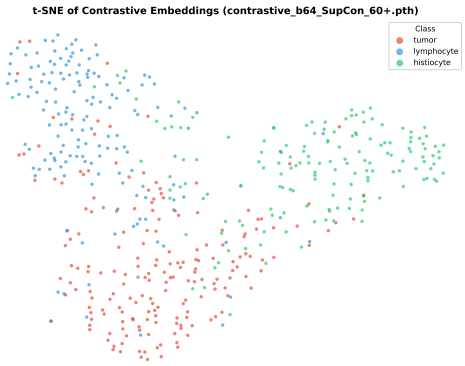
Furthermore, Table 4 shows the intra- and inter-class cosine similarity for both approaches. For the unsupervised SimCLR model, the intra-class similarity is remarkably low (≈ 0.03), suggesting that the encoder treats most image patches as unique instances with nearly orthogonal feature vectors. While the inter-class similarity is also near zero, the lack of intra-class separation indicates a latent space that is not yet discriminative of tissue types.

In contrast, the SupCon model shows a significant increase in intra-class similarity, particularly for Lymphocytes (0.5064). This suggests that the supervised objective successfully forced the model to collapse various augmented views of the same nuclei type into a tighter region of the hypersphere. Although the inter-class similarity also increased slightly (likely due to the global clustering of the entire dataset), the wide margin between intra- and inter-class values for SupCon confirms the more successful linear separation of classes than the unsupervised ap-

proach.



(a) SimCLR (Unsupervised) Contrastive Learning



(b) Supervised Contrastive Learning

Figure 8. t-SNE visualisation of each contrastive learning method.

Table 4. Evaluation of latent space quality using cosine similarity.

	SimCLR		SupCon	
Class	**Intra-class ↑**	**Inter-class ↓**	**Intra-class ↑**	**Inter-class ↓**
Tumor	0.0237	0.0036	0.4175	0.1093
Lymphocyte	0.0260	0.0018	0.5064	0.0942
Histiocyte	0.0373	-0.0025	0.3253	0.0214

Evaluation on Downstream Classification Each of the contrastive pre-trained models were then used as the frozen encoder layers of the classification network. The network trained using the unsupervised contrastive learning (SimCLR) approach achieved a test accuracy of 64.79%, while the network trained using the supervised contrastive learning (SupCon) approach achieved a test accuracy of 69.64%. Both networks performed below the baseline of 70.83%.

These results suggest that while the SupCon approach did produce better feature representations for tissue segmentation than the SimCLR approach, neither were informative enough for the classification head to accurately determine classes. Further pre-training of the SupCon network might produce stronger feature representations as it may better learn to discriminate classes.

3.4.3. Comparison

Table 5 summarises the test-set accuracy for all models alongside the provided baseline. The end-to-end EfficientNet-B0 is the strongest performer, while both contrastive approaches fall below the baseline.

Table 5. Nuclei classification performance and parameter efficiency.

Model	**Accuracy**	**Δ Acc.**	**Parameters**
Baseline	0.7083	—	5.0 M
End-to-End	0.7793	+0.0710	4.34 M
SimCLR	0.6479	-0.0604	0.0038 M
SupCon	0.6964	-0.0119	0.0038 M

The end-to-end classifier achieves 77.93% accuracy, a +7.10 percentage point improvement over the baseline, despite using slightly fewer parameters (4.34M vs ∼ 5.0M). This confirms that end-to-end training with careful dataset construction is sufficient to produce strong nuclei classification performance. In contrast, both contrastive models use only 3,843 trainable parameters during the classification phase, since the encoder is frozen and only the linear head is updated. This is a reduction of more than three orders of magnitude relative to the baseline, which is notable in terms of parameter efficiency even though accuracy is lower.

The SimCLR model achieves 64.79% accuracy, falling −6.04 percentage points below the baseline. The SupCon model improves upon this with 69.64%, but still falls −1.19 percentage points short. This gap is consistent with the latent space analysis in Table 4. SimCLR produces near-zero intra-class cosine similarities (≈ 0.03), indicating that the encoder treats each patch as an almost entirely unique instance, yielding a feature space that is insufficiently discriminative for linear classification. SupCon substantially improves intra-class cohesion, particularly for Lymphocytes (0.5064), and the t-SNE visualisations in Figure 8 confirm more compact, class-specific clusters. Nevertheless, inter-class similarity also increases slightly under SupCon and class boundaries remain imperfect, which explains why the linear head still cannot match the end-to-end model.

The core reason for this performance gap is likeyl the frozen encoder. As discussed in Section 2.3, frozen encoders learn representations that are not specifically optimised for the downstream task, resulting in poorer final performance. This limitation is compounded by the contrastive pre-training dataset being significantly smaller than the supervised training set (3,000 vs 7,500 patches), reducing the encoder's exposure to the full range of variation across the three nuclei classes. Fine-tuning the encoder with a reduced learning rate following contrastive pre-training, as in the unfrozen condition of Task 1, would likely have reduced this gap.

In summary, end-to-end supervised training remains the most effective strategy for this task. Contrastive pre-training produces an informative latent structure, particularly under the supervised approach, but using the frozen encoder as a fixed feature extractor limits downstream accuracy. These results suggest that contrastive pre-training is most valuable as an initialisation strategy rather than a fixed feature extractor.

4. Team Responsibilities

- **Member 1:** Task 1 Dataset Investigation, Task 1 Part b, Task 2 Part b
- **Member 2:** Task 2 Dataset Creation, Task 1 Part a, Task 2 Part a

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[4] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

[5] Miguel López-Pérez, Pablo Morales-Álvarez, Lee A. D. Cooper, Christopher Felicelli, Jeffery Goldstein, Brian Vadasz, Rafael Molina, and Aggelos K. Katsaggelos. Learning from crowds for automated histopathological image segmentation. *Computerized Medical Imaging and Graphics*, 112:102327, 2024.

[6] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[8] Mark Schuiveling. Melanoma histopathology dataset with tissue and nuclei annotations, 2025.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[10] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sébastien Ourselin, and M. Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer, Cham, 2017.

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[12] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[13] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[14] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[15] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

[16] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.